

Appl. No. 09/886,855
Amdt. dated August 25, 2005
Reply to Office Action of June 6, 2005

Remarks

The present amendment responds to the Official Action dated June 6, 2005. The Official Action rejected claims 1, 15, 16, and 18 under 35 U.S.C. §101 as purportedly directed to non-statutory subject matter. Claims 1, 10-15, 19, and 28-33 were rejected under 35 U.S.C. §103(a) based on Faraboschi et al. U.S. Patent No. 5,930,508 ("Faraboschi") in view of Roediger et al. U.S. Patent No. 6,305,014 ("Roediger"). Claims 2-9 and 20-27 were rejected under 35 U.S.C. §103(a) based on Faraboschi and Roediger in view of McKinsey et al. U.S. Patent No. 6,675,380 ("McKinsey"). Claims 18 and 36 were rejected under 35 U.S.C. §103(a) based on McKinsey in view of Faraboschi and further in view of G.J. Chaitin, *Register Allocation & Spilling Via Graph Coloring*, ACM, 1982 ("Chaitin"). Claims 16-17 and 34-35 were rejected under 35 U.S.C. §103 based on McKinsey in view of Faraboschi. These grounds of rejection are addressed below.

Claims 1, 2, 15, 16, 18, 19, 34, and 36 have been amended to be more clear and distinct. In particular, claims 1 and 19 have been amended to clarify the term "lifetime." Support for this claim amendment can be found, for example, at page 11, lines 30-32 of the present specification. Claims 16 and 34 have been amended to clarify that live-in and live-out sets are being determined from a control graph. Claims 18 and 36 have been amended to clarify that the claimed interference graph is derived from a control graph. Support for these claim amendments can be found, for example, at page 12, lines 8-30. Claims 1-36 are presently pending.

Appl. No. 09/886,855
Amdt. dated August 25, 2005
Reply to Office Action of June 6, 2005

Sec. 101 Rejection

Although the Applicants do not acquiesce to this rejection, claims 1, 15, 16, and 18 have been amended to include the phrase "computer implemented method" as suggested by the Examiner.

The Art Rejections

As addressed in greater detail below, Faraboschi, Roediger, McKinsey, and Chaitin do not support the Official Action's reading of them and the rejections based thereupon should be reconsidered and withdrawn. Further, the Applicants do not acquiesce in the analysis of Faraboschi, Roediger, McKinsey, and Chaitin made by the Official Action and respectfully traverse the Official Action's analysis underlying its rejections.

Claims 1, 10-15, 19, and 28-33 were rejected under 35 U.S.C. §103(a) based on Faraboschi in view of Roediger. Faraboschi describes compacting a very long instruction word in a processor by eliminating no operation (NOP) codes from the VLIW instruction. Faraboschi, col. 2, lines 66-67. The NOP codes in a VLIW instruction correspond to operation codes for functional units that are not needed during execution of the VLIW instruction. Faraboschi, col. 1, lines 52-57. To this end, Faraboschi's approach involves identifying each word of an instruction that does not contain a NOP code, generating a dispersal code for each identified word, generating a delimiter code for each identified word and storing each identified word along with the corresponding dispersal and delimiter code. Faraboschi, col. 3, lines 1-10. Unlike the present invention, Faraboschi addresses a totally different problem of compacting VLIW

Appl. No. 09/886,855
Amdt. dated August 25, 2005
Reply to Office Action of June 6, 2005

instructions by eliminating NOP codes from an instruction. By contrast, the present invention addresses techniques for allocating VLIW instructions to VLIW instruction memory (VIM) regardless of whether the VLIW instruction contains a NOP code or not. See page 2, lines 8-10 of the Specification, for example, for a discussion of VIM allocation.

Claim 1 recites "allocating at least some of the plurality of VLIW instructions to VIM." See also claim 19. With respect to claim 15, the Official Action attempts to equate eliminating NOP codes from a VLIW as taught by Faraboshchi with the objective of the present invention of reducing redundant loads of very long instruction word instruction memory. Applicants respectfully disagree. NOP codes are used under the assumption that each execution unit should perform an operation on every processor cycle. Faraboshchi, col. 1, lines 41-43. Rather than merely eliminating the storage of NOP codes in memory, claim 15 reduces redundant loading of a load VLIW (LV) instruction by "selecting a load VLIW (LV) instruction in a current node; and placing the LV instruction in a new node which is closer to a program start node if an execution frequency of the new node is lower than an execution frequency of the current node, and if a maximum number of VIM lines is not exceeded." Faraboshchi's disclosure is silent with respect to selecting and placing of an LV instruction as presently claimed. See also claim 33. Furthermore, the Official Action admits that Faraboschi does not explicitly disclose determining a lifetime of each of said plurality of VLIW instructions and relies on Roediger accordingly.

Roediger does not cure the deficiencies of Faraboschi. Roediger addresses an instruction scheduler in an optimizing compiler by determining the lifetimes of fixed registers. Roediger, col. 1, lines 62-65. During the compilation, symbolic registers are mapped to fixed registers

Appl. No. 09/886,855
Amdt dated August 25, 2005
Reply to Office Action of June 6, 2005

based on the lifetime of the fixed registers. However, Roediger does not address "allocating at least some of the plurality of VLIW instructions to VIM based on the lifetime of said plurality of VLIW instructions" as claimed in claim 1. (emphasis added)

The Official Action suggests that Roediger and Faraboschi should be combined "to provide faster execution of [a] computer program." There is no suggestion to combine these two references in the manner outlined by the Examiner. Roediger and Faraboschi cannot simply be combined to meet the presently claimed invention. By way of example, the combination of the compression of VLIW instructions containing NOP codes as taught by Faraboschi with the determination of the lifetime of fixed registers as taught by Roediger simply does not specify how to allocate VLIW instructions to VIM. Referring to the Specification at page 2, lines 17-21, unlike the Faraboschi and Roediger combination, the present invention advantageously addresses an approach in which a small VIM size may be used even when a software application demands a number of VLIWs larger than can be fit into the physical VIM size of a processor.

In the Examiner's Response portion of the Official Action, the Examiner relies upon the disclosure of page 2, line 5-15 of the present specification as the basis for his position that eliminating NOP codes from instructions is similar to efficiently allocating VLIW instructions to VIM as claimed. Applicants respectfully disagree. The Examiner apparently misreads the cited text. At page 2, lines 5 -7, the cited text reads "indirect VLIW not only avoids the explicit storage of non-operational place holders (nop instructions), but also enables code compression by storing more than one non-overlapping VLIW in the same VIM line." (emphasis added)

Although the present specification provides a solution to the problem of eliminating the storage

Appl. No. 09/886,855
Amdt. dated August 25, 2005
Reply to Office Action of June 6, 2005

of NOP instructions, the present invention additionally addresses the problem of deciding when a VLIW instruction should be loaded into a VIM line. Specification, page 2, lines 8-10. To this end, the claimed invention addresses this separate problem by “allocating at least some of the plurality of VLIW instructions to VIM based on the lifetime of said plurality of VLIW instructions,” as presently claimed in claim 1. See also claim 19 as presently amended. Claims 15 and 33 address this problem by strategically “placing the LV instruction in a new node which is closer to a program start node if an execution frequency of the new node is lower than an execution frequency of the current node, and if a maximum number of VIM lines is not exceeded.”

Furthermore, the Examiner’s Response also notes that Roediger is relied upon for purportedly suggesting the limitation “determining a lifetime of each of said plurality of VLIW instructions.” As discussed above and in the previous response, Roediger merely describes determining the lifetimes of fixed registers. The lifetime of a fixed register, as defined in Roediger at col. 3, line 66 – col. 4, line 6, is “a set of instructions that operate with that register, either by ‘defining’ the register (i.e., assigning it a value), or ‘using’ the register (i.e., reading the current value in the register and using it in a computation). The lifetime contains a single defining instruction, and zero or more subsequent instructions that use the value stored by the defining instruction.” Unlike the lifetime of a fixed register storing data used by instructions, the present invention defines a lifetime of an instruction itself. Claims 1 and 19, as presently amended, recite “the lifetime of a VLIW instruction including the interval of time between loading the VLIW instruction to VIM and the last time the VLIW instruction is executed.”

Appl. No. 09/886,855
Amdt. dated August 25, 2005
Reply to Office Action of June 6, 2005

Roediger does not teach and does not suggest determining the lifetime of a VLIW instruction as presently claimed.

Moreover, the Examiner's Response implies that Applicants' arguments do not address the combination of references. Applicants respectfully disagree. As discussed above and in the previous response, if the compression of VLIW instructions containing NOP codes as taught by Faraboschi were combined with the determination of the lifetime of fixed registers as taught by Roediger, the combination would still fail to address how to allocate VLIW instructions to VIM and would not meet the claims as presently amended.

Claims 2-9 and 20-27 were rejected under 35 U.S.C. §103(a) based on Faraboschi and Roediger in view of McKinsey. McKinsey fails to cure the deficiencies of Faraboschi and Roediger. Since claims 2-9 depend from and contain all the limitations of claim 1, as presently amended, claims 2-9 distinguish from the references in the same manner as claim 1. Since claims 20-27 depend from and contain all the limitations of claim 19, as presently amended, claims 20-27 distinguish from the references in the same manner as claim 19.

Furthermore, McKinsey addresses a path speculating instruction scheduler. McKinsey, Abstract. With McKinsey's system, instructions may be executed out of order to improve the performance of a processor. In so doing, McKinsey's system includes creating control flow graphs made up of blocks of instructions. The control flow graphs are utilized to determine whether a block of instructions can be executed in the same machine cycle, or may be executed in different machine cycles depending on the available resources in the processor and data dependencies between instructions. McKinsey, col. 5, lines 29-42. Unlike the present invention,

Appl. No. 09/886,855
Amdt. dated August 25, 2005
Reply to Office Action of June 6, 2005

McKinsey's utilization of control graphs does not address allocation of a VLIW instruction to VIM.

Unlike McKinsey, the present invention utilizes a control graph to determine the lifetime of a VLIW instruction. In general, a VLIW instruction has to be loaded with a load instruction into VIM memory, for example, and subsequently executed with an execute instruction. To address this special behavior of VLIW instructions, the present invention determines three graphs; a control graph, a VLIW flow graph, and an interference graph. Claim 2, for example, clarifies the step of determining the lifetime of a VLIW instruction when it recites "determining a control flow graph for the input source program containing said plurality of VLIW instructions; determining a VLIW flow graph for said plurality of VLIW instructions; and determining a VLIW interference graph." McKinsey does not disclose and does not make obvious determining the lifetime of a VLIW instruction by "determining a control flow graph for the input source program containing said plurality of VLIW instructions; determining a VLIW flow graph for said plurality of VLIW instructions; and determining a VLIW interference graph," as claimed in claim 2.

Claims 18 and 36 were rejected under 35 U.S.C. §103(a) based on McKinsey in view of Faraboschi and further in view of Chaitin. Claim 18, as presently amended, recites the step of "determining live-out sets for the plurality of nodes, the live-out sets and the control graph defining a VLIW flow graph." While McKinsey discloses a control graph, also referred to as a dependence graph, for instructions which do not have VLIW instruction behavior as discussed above, McKinsey does not teach and does not suggest defining a VLIW flow graph from a

Appl. No. 09/886,855
Amdt. dated August 25, 2005
Reply to Office Action of June 6, 2005

control graph and “determining an interference graph from the VLIW flow graph” as presently claimed in claims 18 and 36. Furthermore, McKinsey does not teach and does not suggest “inserting an undirected edge into the interference graph between two VLIW nodes,” as presently claimed in claims 18 and 36. (emphasis added). The only graphs disclosed in McKinsey require directed edges.

Faraboschi fails to cure the deficiencies of McKinsey. Faraboschi is relied upon for compacting VLIW instruction. As discussed above, the present invention is not merely compacting an individual VLIW instruction by removing a NOP code. Rather, claims 18 and 36 address determining “interference between indirect very long instruction word (VLIW) instructions.”

Chaitin fails to cure the deficiencies of McKinsey and Faraboschi. Chaitin is relied upon for coloring VLIW graph nodes. Chaitin addresses graph coloring techniques on a graph obtained from “well-known optimizing compiler techniques to do a global data-flow analysis.” Chaitin, p. 99, left col., lines 19-21.

Unlike Chaitin, the present invention addresses coloring the VLIW graph nodes of a VLIW graph which is not a data-flow graph. The VLIW graph was obtained from determining the live-out sets of the control graph. Claim 18, as presently amended, recites “determining live-out sets for the plurality of nodes, the live-out sets and the control graph defining a VLIW flow graph.” Chaitin does not teach and does not suggest “coloring the VLIW graph nodes such that adjacent VLIW nodes are colored in different colors and each color corresponds to a different VIM line,” as presently claimed in claims 18 and 36.

Appl. No. 09/886,855
Amdt. dated August 25, 2005
Reply to Office Action of June 6, 2005

Even if there was a suggestion to combine the teachings of McKinsey, Faraboschi, and Chaitin, such combination fails to meet the elements of these claims as presently amended. The combination of the control graph of McKinsey, the compressed VLIW instruction of Faraboschi, and the coloring of a data flow graph of Chaitin does not teach and does not suggest “determining live-out sets for the plurality of nodes, the live-out sets and the control graph defining a VLIW flow graph; determining an interference graph from the VLIW flow graph ... inserting an undirected edge into the interference graph between two VLIW nodes ... [and] coloring the VLIW graph nodes,” as presently claimed in claim 18.

Claims 16-17 and 34-35 were rejected under 35 U.S.C. §103 based on McKinsey in view of Faraboschi. The deficiencies of McKinsey and Faraboschi discussed above apply here as well. Claim 16, as presently amended, recites “determining a live-in set and a live-out set of VLIW instructions for each node in the control graph to define a VLIW flow graph, a live-in set for a node comprises the VLIW instructions that are used in the node, a live-out set for a node comprises a union of live-in sets of successor nodes, the determining step further including solving VLIW flow equations for the live-in set and the live-out set.” The Official Action relies on col. 10, lines 7-9 of McKinsey to purportedly suggest determining the liveness of indirect very long instruction word (VLIW) instructions. Applicants respectfully disagree with this reading of McKinsey. At the cited portion of text, the “live on exit” value is computed when a specific instruction requests so and is “used by instructions outside the region R.” McKinsey fails to discuss how or under what conditions instructions outside the region R use this “live on exit” value.

Appl. No. 09/886,855
Amdt. dated August 25, 2005
Reply to Office Action of June 6, 2005

Unlike McKinsey and Faraboschi, the present invention determines liveness of a VLIW by determining a control flow graph and determining live-in sets and live-out sets of VLIW instruction for each node in the control graph. McKinsey and Faraboschi, separately or in combination, do not teach and do not suggest "determining a live-in set and a live-out set of VLIW instructions for each node in the control graph to define a VLIW flow graph, a live-in set for a node comprises the VLIW instructions that are used in the node, a live-out set for a node comprises a union of live-in sets of successor nodes, the determining step further including solving VLIW flow equations for the live-in set and the live-out set," as presently claimed in claim 16.

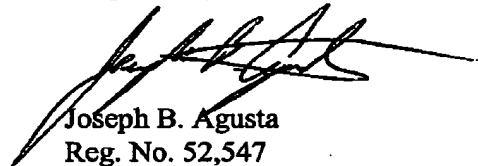
The relied upon references fail to recognize and address the problem of allocating VLIW instruction memory in the manner advantageously addressed by the present claims. The claims as presently amended are not taught, are not inherent, and are not obvious in light of the art relied upon.

Appl. No. 09/886,855
Amdt. dated August 25, 2005
Reply to Office Action of June 6, 2005

Conclusion

All of the presently pending claims, as amended, appearing to define over the applied references, withdrawal of the present rejection and prompt allowance are requested.

Respectfully submitted,



Joseph B. Agusta
Reg. No. 52,547
Priest & Goldstein, PLLC
5015 Southpark Drive, Suite 230
Durham, NC 27713-7736
(919) 806-1600 x3